University of Glasgow

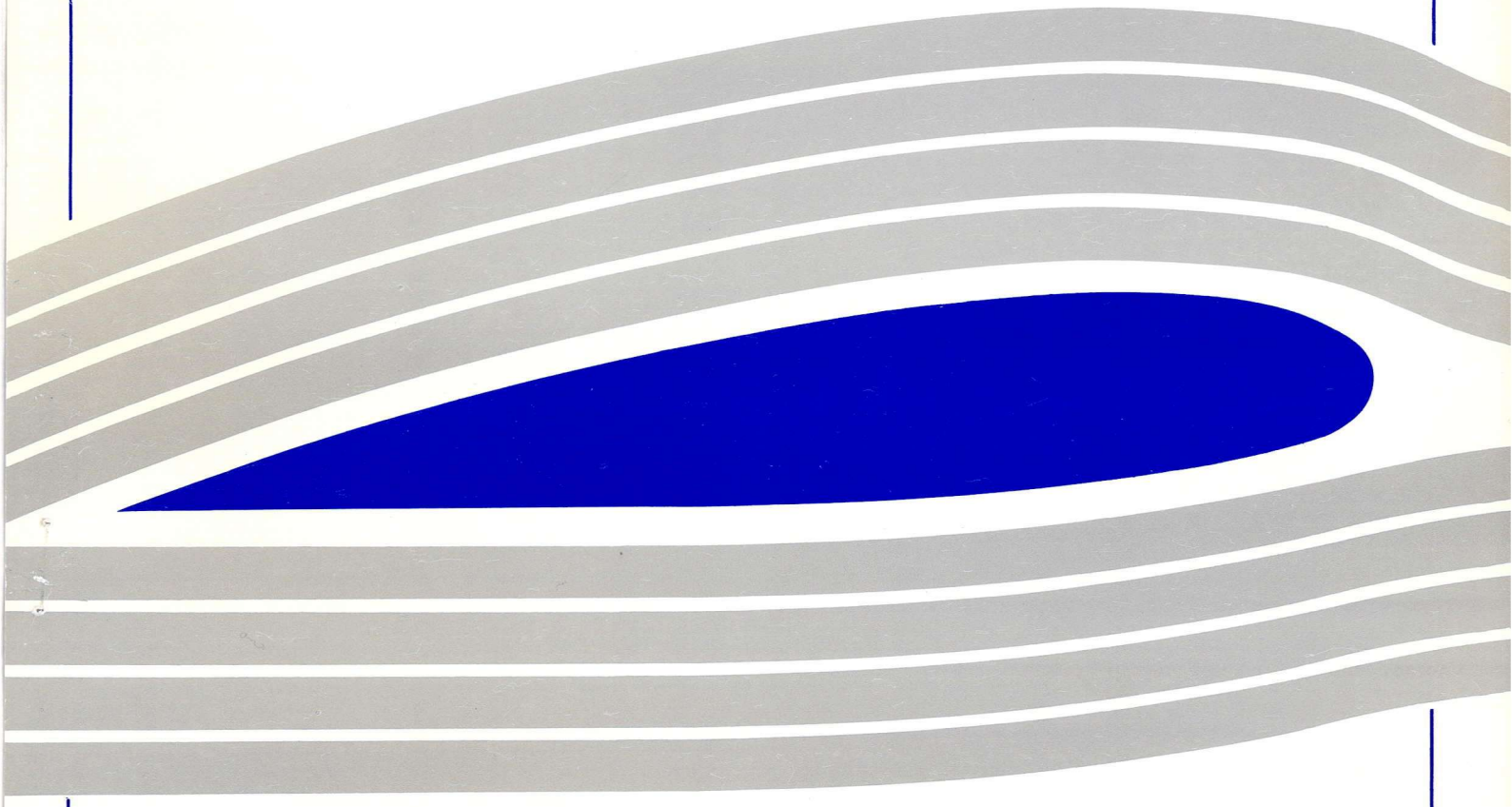
DEPARTMENT OF

AEROSPACE ENGINEERING

Parallelization of explicit and fully implicit Navier-Stokes solutions for compressible flows

X, Xu, N, Qin, B E Richards

GU Aero Report 9236 July 1992



Parallelization of explicit and fully implicit Navier-Stokes solutions for compressible flows

X, Xu, N, Qin, B E Richards
GU Aero Report 9236 July 1992

Department of Aerospace Engineering
University of Glasgow
Glasgow
G12 8QQ

Communication to Parallel CFD '92,
New Brunswick, New York, May 1992.

Parallelising explicit and fully implicit Navier-Stokes solutions for compressible flows

X.Xu, N.Qin and B. E. Richards.

Department of Aerospace Engineering, University of Glasgow, G12 8QQ, UK.

Abstract

The paper describes two studies involved with the parallelisation of algorithms for the numerical calculation of hypersonic viscous flows over generic vehicle configurations by solving the Navier-Stokes equations. One involved a scalable explicit formulation that achieved high parallel efficiency on 32 processors of an Intel iPSC860 Hypercube when calculating the 3-dimensional flow over a blunt delta wing at high incidence, the other involved a fully implicit formulation using a Newton-like procedure with a GMRES solver with pre-conditioning when high efficiency was achieved when run on 8 transputers of a Meiko Computing Surface computer in order to calculate the flow over a cone at high incidence. This latter approach, although more complex, has potential in providing more rapid convergence characteristics, hence efficiency, than the explicit scheme. High order upwind discretisation was used in each case in order to achieve high resolution of important shock and viscous phenomenon within the flow field. The work reported contributes to the aim of a wider programme of work, a summary of which is included, to provide the computational tools to calculate accurately and efficiently steady and unsteady viscous compressible flows over complex aerospace configurations.

1. INTRODUCTION

Following many studies by the Computational Fluid Dynamics (CFD) Team at Glasgow University on the adaption of current state of the art CFD techniques towards predicting hypersonic viscous flows, it is apparent that with increasing complexity of application, there are difficulties in convergence to sufficient accuracy using explicit and approximate implicit schemes without using massive amounts of time on even the most powerful of national super-computing facilities. A brief commentary on these studies is given in the next section. There appears to be a contemporary train of thought that with the advent of more powerful computers these difficulties will be easily overcome and that the present algorithms will be sufficient. This view however ignores several general points. The past history of rapid developments in CFD was achieved not only by the availability of powerful hardware but also by the development of clever new algorithms. Secondly, the predictions of continuing rapid advances in the future have assumed that the opportunities offered by new architectures will be fully grasped. It is with this in mind that the team has explored, for hypersonic and transonic applications, the characteristics of CFD discretisation techniques and algebra solvers and from this experience either chosen those that give the best accuracy or developed new techniques, such as acceleration procedures or algebra solvers, where these are deficient. Both vector processors and parallel architectures have been used in these developments. The developments to date in parallel computing will be the main focus of this paper.

2 BACKGROUND

In this work a cell-centred finite volume formulation is used to reduce sensitivity to the highly stretched/skewed meshes generated in the types of problem to be tackled. Structured grids are used for spatial discretisation, but unstructured grids are not precluded. An upwind approach, namely Osher's flux difference splitting (FDS) method, has been chosen to be used following demonstrations [1,2] that it provides low numerical dissipation in dealing with viscous/ turbulent flows. By using a third order MUSCL scheme, good resolution of shocks, boundary layers and wakes is achieved so that the number of grid meshes can be reduced for a particular problem. These demonstations have been done on the corner flow problem of two intersecting 8 degree wedges at right angles and 30° sweep and at M = 12.75, Re= 5 x 10^6/m, T = 38.73 K, wall T = 300 K [1] and the Tracy [3] flow of a 7° half-angle cone at 0° angle of attack, M = 7.95, Re = 4.2 x 10^6m, T = 55.4 K, wall T = 309.8 K [2].

The high order explicit finite volume Osher FDS approach has been deployed to numerically solve the fully 3D-NS equations. An explicit multi-stage Runge-Kutta method was employed in the iterative time integration in the 3D-NS part using local time stepping. The full NS model needs to be used for forebody calculations and for separated flows due to upstrem influence, but many afterbody flows including viscous effects can be tackled adequately by using 3D-P(arabolised)NS modelling. In a 3D-PNS code developed, the unsteady term and the viscous derivatives in the streamline direction were omitted and the streamwise pressure gradient terms treated using Vigneron's approach. In the resulting marching method an implicit treatment in the marching plane was employed which required the iterative solution of a non-linear system at each station. A combined programme has been used successfully [4] to calculate the flow field around a generic vehicle configuration representing the ESA Hermes which was in the shape of a blunt delta wing as illustrated in Figures 1 (the grid used) and 2 (results of the iso-Mach no. lines). Despite the success in calculation, however, overall convergence was slow. This is attributed to solution dependence on cell Reynolds number, necessitating the use of the very stretched grids to resolve the high gradients in thin shear layers near walls and the use of the sophisticated high order upwind scheme.

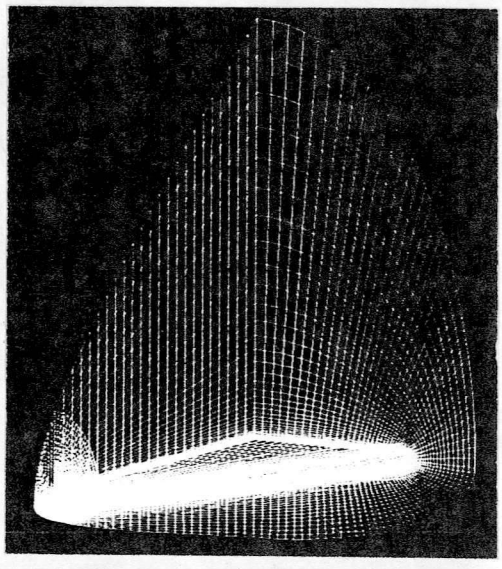


Fig 1. Grid around blunt leading-edged delta-wing. Symmetry plane and rear surface.

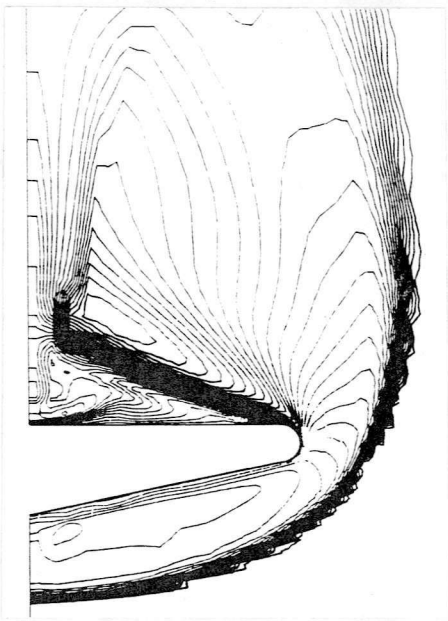


Fig 2. Mach number contours at cross sectional plane at 50% chord of delta wing.

Current means of overcoming this convergence difficulty have been through developments in multigrid methods aimed mainly (but not exclusively) at explicit methods and through

implicit time marching approaches which have been shown to achieve a degree of success in Euler solvers for inviscid compressible flows. However these developments have been disappointing for complex problems in that the efficiency of multigrid techniqes have been found to be significanly reduced on these highly stretched grids. Furthermore implicit procedures involve linearization which, if exact, implies analytical calculation of the Jacobian of the non-linear system. This feature then creates difficulties in dealing with the inherent non-linearity of the Navier-Stokes equations using high resolution schemes. The result is that the majority of implicit methods use an approximate linearization, using for example only first order upwind inviscid terms in the implicit operator, which generally results again in poor convergence for Navier-Stokes solvers in this case because of the unbalanced left and right hand sides.

A fully implicit approach enables virtual elimination of the stability problems that cause these comparatively slow convergences in explicit and approximate implicit approaches which bear also upon their ability to tackle efficiently complex steady and time-accurate unsteady flows. The application of the sparse finite difference Newton (SFDN) and sparse quasi-Newton (SQN) methods, developed for CFD applications at Glasgow [5], to the evaluation of the Jacobian, provides the ability to tackle highly non-linear problems generated for example by transonic behaviour, turbulence and real gas effects and provides fast convergence (quadratic or superlinear, respectively).

Adopting these approaches, however, does pose the dual problem of dealing efficiently with the solution of the resulting large 13-point (for 2-d, 21 for 3-d cases) ill-conditioned sparse non-symmetric linear system and providing for the large memory required in this approach. The efficient linear solver recently devised in this continuing research [6] to tackle the first problem is a multi-level iterative method using the generalised minimum residual (GMRES) approach with a diagonalised preconditioner and a damping factor (α, giving rise to the name α-GMRES solver).

The overall scheme has been tested, with excellent comparison, on the case due to Tracy [3] of the laminar flow over a 10^o cone at 24^o incidence in a Mach 7.95 flow with Reynolds number of 4.2×10^6, wall temperature of 309.8 K and freestream temperature of 55.4 K. The flow is a demanding case including massive flow separation, bow and flow embedded shocks and very high temperature gradient in the windward boundary layer as illustrated in Figure 3.

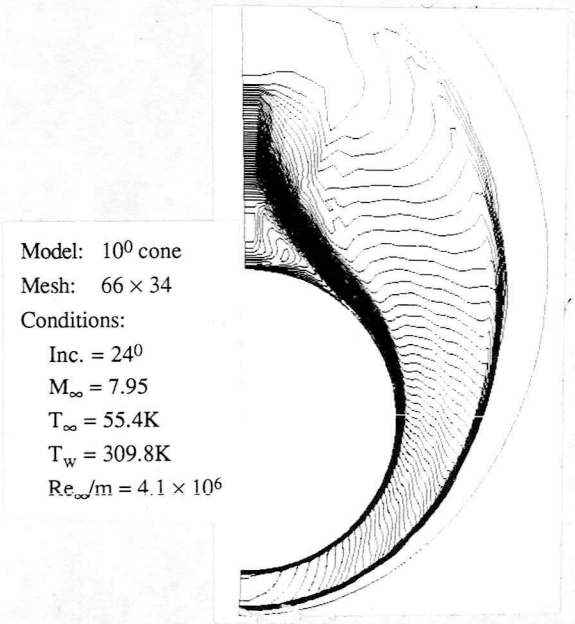


Fig 3. Flowfield temperature contours at a cross section of the cone.

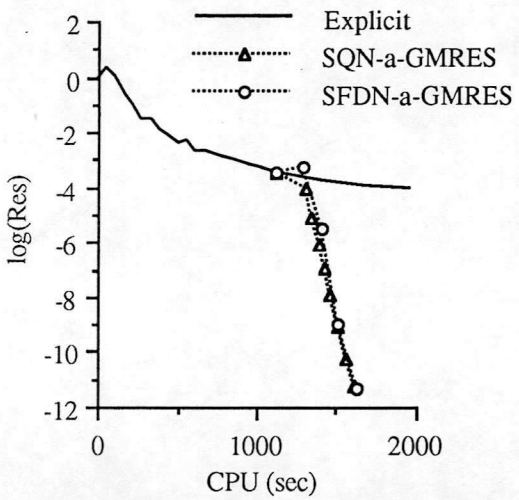


Fig 4. Convergence of SFDN and SQN methods using α-GMRES solver for NS solution.

However since the flow is nearly conical then it can be well modelled by the locally conical Navier-Stokes equations thus reducing the dimensions of the problem by one. Mesh sizes of 34 x 34 and 66 x 34 have been used. Fig. 4 demonstrates using an IBM RS6000, the fast convergence at low processor cost of the SFDM-α-GMRES and SQN-α-GMRES methods, once the calculation had been started by an explicit method. This starting solution shown used the time dependent approach with Runge-Kutta integration and local time stepping. This approach is robust for starting the solution from freestream conditions, but slow in convergence. An approximate eigenvalue analysis [7] using Arnoldi's method revealed the favourable effects of preconditioning and damping on the eigenvalue spectra which explains the successful convergence characteristics of the GMRES method.

In the overall research programme numerical solutions of the Reynolds' averaged 3-d N-S equations have been made using the Baldwin-Lomax turbulence model [8]. The non-equilibrium models of Johnson-King and K~ε models which have been deployed before in earlier 2-d studies are presently being incorporated. Some fast subroutines for including equilibrium air [9] and eventually non-equilibrium and frozen air for very high gas energies have been developed to tackle high enthalpy cases. Additions of this nature to a Navier-Stokes model make the systems to be solved more non-linear which thus make solution convergence even more difficult to achieve.

It is to explore this magnitude, in terms of processing and also particularly memory requirements, of problem associated with viscous aerodynamic design support, that algorithms are now being developed by the Research Team to make use of the new generation of scalable distributed memory parallel computers. The paper then describes two pieces of work involved with the parallelization of algorithms for the numerical calculation of hypersonic viscous flows over generic vehicle configurations. Efficient message passing and balanced use of the multi-processors has been a theme of the research. Chapter 3 describes a scalable explicit formulation that achieved high parallel efficiency on 32 processors of an Intel iPSC860 Hypercube for solving the full 3-dimensional Navier-Stokes equations. Chapter 4 describes a fully implicit formulation using a Newton-like procedure and a GMRES solver with pre-conditioner in which high parallel efficiency was achieved on 8 transputers of a Meiko Computing Surface in solutions of the locally conical Navier-Stokes equations.

3. PARALLELISING THE EXPLICIT FINITE VOLUME SOLVER

In this section, we present some aspects of parallelisation of the explicit three dimensional finite volume Navier-Stokes solver on a hypercube parallel computer. Aspects of the parallelisation of the implicit operator will be dealt with in the next section.

Parallel processing introduces several concepts into the design of the software, that are unique to parallel programming. These are the concepts of "load balancing", "communication" and "scaling". After a brief description of the hypercube parallel computer in Sec.3.1, we discuss the parallelisation of the three dimensional finite volume Navier-Stokes solver in the following sections according to these basic principles.

3.1 The Intel iPSC/860 Hypercube Computer

The INTEL iPSC/860 hypercube is a parallel computer that makes concurrent computation available at a relatively low cost. It consists of a set of independent i860 processors, each with its own memory and capable of operating on its own data. Each processor has its own program to execute and processors are linked together by communication channels. Due to the "hypercube" connectivity, the number of the processors is always of the form 2^d, where d is an integer and known as the dimension of the hypercube.

The hypercube architecture has a number of desirable features. The network provides a good balance between the requirement for a high degree of connectivity between the processing nodes and the engineering requirements of ease of construction and minimal cost. In addition,

the hypercube provides a "fixed" architecture (in the conceptual sense), which simplifies the design of software and allows the rescaling of an algorithm to larger numbers of nodes to occur in a simple way.

Communication between nodes is handled by generic routines supplied by the manufacturer. These routines permit the user to send and receive messages of specified length between specified nodes.

3.2 Domain Decomposition and Load Balance

Domain decomposition is a technique well suited for CFD problems, where the fundamental data structures can be decomposed by splitting up the physical domain of the computation. For the present three dimensional Navier-Stokes calculation, the domain decomposition has been achieved through subdividing the three dimensional block into slices in the streamwise direction.

The discretisation used in the Navier-Stokes solver requires a 21-point stencil. At the interfaces of each subdomain, the information at two grid surfaces in the neighbouring subdomains is required to complete the update of the solution at all the cell centres in the current subdomain. Therefore communication between nodes has to be arranged.

It is important that the total workload is equally distributed among the nodes to avoid wasting the computing resources of the system. Thus the above subdivision is carried out in such a fashion that each subdomain will have the same amount of grid points and therefore the same amount of workload apart from a small difference due to the two boundary slices.

3.3 Scalability

Another of the programming principles is to make the code scalable; that is, to program it so it can be executed independent of the number of nodes currently available to be used. A special effort has been made to make the parallel code scalable. The code adjusts the array size according to the size of the subcube allocated to the node and is made independent of the size of the cube.

3.4 Accuracy and Efficiency

It is obvious that the accuracy and the iterative convergence of the sequential code should be maintained with the current parallel approach. To study the efficiency of the parallel approach, computation has been carried out to solve the hypersonic viscous flow around a blunt delta wing related to the European Hermes space programme using a C-O grid of the size 65x33x33 similar to that in Fig 1 (earlier). The results are identical to that illusttrated in Fig 2 (earlier). The computing time, speedup and efficiency are shown in the following table using 1 to 32 processors.

d	No. of Processors (p)	Computing Time (T_p, Second/Iteration)	Speedup ($S_p=T_1/T_p$)	Efficiency ($E_p=S_p/p$)
0	1	129.5	1.00	1.00
1	2	65.3	1.98	0.99
2	4	33.0	3.92	0.98
3	8	17.6	7.35	0.91
4	16	9.0	14.39	0.90
5	32	5.4	23.98	0.75

In the present computation, the surface communication time is comparatively shorter than the computation time in the subdomain, which yields in a relatively high performance on the medium grain parallel computer. It is clear from the table that the efficiency reduces as the number of the processor increases. This is because in the present computation the total computation per iteration is constant and the communication time for each subdomain (or processor) is also constant since the same amount of surface data has to be sent or received no

matter how large or small is the subdomain. However the computation time in each subdomain is shorter if the subdomain is smaller when using more processors. Thus the ratio of the computation time to communication time becomes smaller and smaller if more and more processors are used. The performance given in the table above is compatible with a communication time, T_c, of around 1.1 seconds. It can be seen that T_c is small when compared with a computing time of 65.3 seconds for the 2 processor case, but significant when compared with 5.4 seconds for the 32 processor case.

The number of iterations required for reasonable convergence in this implicit approach is high (about 8000 iterations). Multigrid acceleration will reduce processor time, however the authors feel that a more efficient approach is through the parallelised fully implicit scheme outlined in the following section.

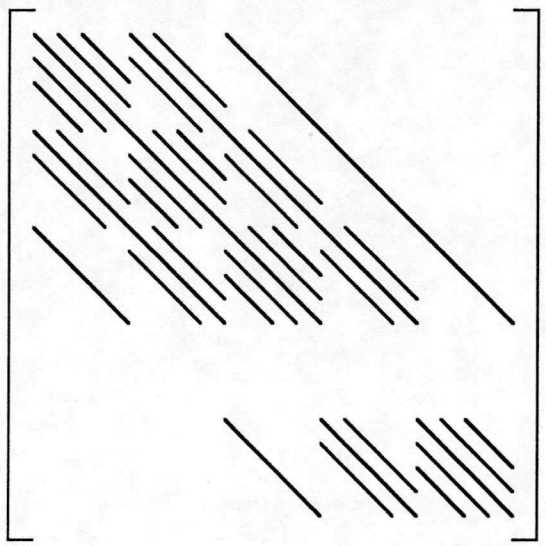
4. PARALLELISING THE FULLY IMPLICIT SOLVER

In the Background Section in which the fast convergence characteristics of the fully implicit approach was illustrated we also referred to the large memory requirements of the method. This memory problem that becomes critical for very large grids is being tackled through the use of distributed memory multi-processors. These provide the features of fast processing, high memory and low cost enabling these large problems to be tackled. Parallelisation is then used both to evaluate the Jacobian and solve the linear system

First we discuss the linear solver. We denote the linear system by

$$A\,x = b \qquad (4.1)$$

where the structure of A depends on the spatial discretisation scheme used. Typically we consider the following system resulting from a second or third order high resolution scheme using a structured grid for a two-dimensional Navier-Stokes solution. The linear system will be a block 13-point diagonal matrix which can be denoted as



4.1. The α-GMRES Method

For the large sparse non-symmetric linear system $A\,x = b$, the α-GMRES method [6] is written as

$$(\alpha I + D^{-1}A)\,x^{n+1} = D^{-1}b + \alpha x^{n}. \qquad (4.2)$$

Given x^n, the above equation is solved for x^{n+1} using the GMRES method [10]. This procedure is continued until the sequence x^n is converged, and the convergent vector is the solution of the original linear system. Here D is a block diagonal matrix of A, I is a unit matrix, and $\alpha > 0$.

4.2. The Parallel α-GMRES Method

It is assumed there are P processors available.

4.2.1. Data Distribution

The matrix A can be written in columns as $A = [\, A^1, A^2, \ldots, A^P \,]$, where A^p are N×L submatrics, p=1,2,...,P. The transposition of vector v can be written as $v^T = [\, (v^1)^T, (v^2)^T, \ldots, (v^P)^T \,]$ where v^p is vector of order L corresponding to A^p, p=1,2,...,P. A^p and v^p are stored in each processor p. The distribution of the matrix data in columns does not increase the data storage compared to the sequential case.

Remark: The L in the order of N×L may be not the same in different processors.

4.2.2. Parallel Algorithm

Let ε_1 be the convergence criterion of the GMRES algorithm and ε_2 be the convergence criterion of the α-GMRES algorithm. In processor p, we perform the following calculations and communications.

Step 1: Initialisation

Set an initial guess x^p_0, we have

$$A^p x^p_0 = \bar{r}^p_0 \quad , \qquad (*)$$
$$r^p = b^p - \bar{r}^p \quad ,$$

and

$$\|r_0\| = \sqrt{\sum_{p=1}^{P} (r^p_0 \, , r^p_0)} \quad . \qquad (**)$$

Let $\delta = \|r_0\|$ and $w^p_0 = x^p_0$.

Remarks:

(*): Matrix-vector multiplication can be generated as follows:

$$A v_i = (A^1, A^2, \cdots, A^P)\left(\begin{pmatrix} v^1{}_i \\ 0 \\ \vdots \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ v^2{}_i \\ \vdots \\ 0 \end{pmatrix} + \cdots + \begin{pmatrix} 0 \\ 0 \\ \vdots \\ v^P{}_i \end{pmatrix} \right) = A^1 v^1{}_i + A^2 v^2{}_i + \cdots + A^P v^P{}_i$$

$$= \begin{pmatrix} * \\ * \\ \vdots \\ * \end{pmatrix} + \begin{pmatrix} * \\ * \\ \vdots \\ * \end{pmatrix} + \cdots + \begin{pmatrix} * \\ * \\ \vdots \\ * \end{pmatrix} \Rightarrow \begin{pmatrix} \bar{v}^1{}_i \\ 0 \\ \vdots \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ \bar{v}^2{}_i \\ \vdots \\ 0 \end{pmatrix} + \cdots + \begin{pmatrix} 0 \\ 0 \\ \vdots \\ \bar{v}^P{}_i \end{pmatrix}$$

where "⇒" indicates the communication of data among different processors to form $\bar{v}_i$. In this way, the task of calculating $A v_i$ for P processors is divided by calculating $A^p v^p_i$ on processor p. This is the main calculation in the GMRES method. The resulting vector $\bar{v}_i$ is

again distributed to the P processors. The only communication required in the calculation is in the formation of $\overline{v}_i$. Due to the sparsity of the matrix A, this communication is only of a limited nature.
(**): Here requires the collection of the partial inner products carried out on each processor.

Step 2: Calculate $B = (\alpha I + D^{-1}A)$ and $c = D^{-1}b$

We can write B in columns as $B = [B^1, B^2, ..., B^P]$, which has the same stencil as matrix A. Parts of D^{-1} are calculated in each processor separately. Thus $D^{-1}A$ can be performed in each processor provided that appropriate communications are arranged. Then the α is added in diagonal elements in each processor so that we have B^p in each processor. $c = D^{-1}b$ can be performed in each processor without any communication.

Step 3: Calculation

Let $e^p_0 = c^p + \alpha w^p_0$ and we have

$$B^p w^p_0 = \bar{f}^p_0 \ ,$$
$$f^p_0 = e^p_0 - \bar{f}^p_0 \ ,$$

and then set

$$\hat{v}^p_1 = f^p_0 \ ,$$

so we have

$$\|f_0\| = \sqrt{\sum_{p=1}^{P} (f^p_0 , f^p_0)} \ .$$

$$v^p_1 = \frac{f^p_0}{\|f_0\|} \ .$$

Let $\delta_1 = \|f_0\|$ in the first iteration.

Step 4: For i=1 to k

$$B^p v^p_i = \overline{v}^p_i \ ,$$

the elements of the Hessenberg matrix are calculated using

$$\beta_{i+1,j} = \sum_{p=1}^{P} (\overline{v}^p_i , v^p_j) \ .$$

We then calculate

$$\hat{v}^p_{i+1} = \overline{v}^p_i - \sum_{j=1}^{i} \beta_{i+1,j} v^p_j \ ,$$

and

$$\|\hat{v}_{i+1}\| = \sqrt{\sum_{p=1}^{P} (\hat{v}^p_{i+1} , \hat{v}^p_{i+1})} \ ,$$

and normalise the base vector as follows

$$v^p_{i+1} = \frac{\hat{v}^p_{i+1}}{\|\hat{v}_{i+1}\|} \ .$$

After k steps, the Hessenberg matrix is

$$H_k = \begin{pmatrix} \beta_{2,1} & \beta_{3,1} & \cdots & \beta_{k+1,1} \\ \|\hat{v}_2\| & \beta_{3,2} & \cdots & \beta_{k+1,2} \\ 0 & \|\hat{v}_3\| & \ddots & \vdots \\ \vdots & \vdots & \ddots & \beta_{k+1,k} \\ 0 & 0 & \cdots & \|\hat{v}_{k+1}\| \end{pmatrix}_{(k+1)\times k} .$$

Step 5: Uses a Q-R algorithm to find y such that

$$\| \delta_2 e_1 - H_k y \| = \min_{y_0 \in R^k} \| \delta_2 e_1 - H_k y_0 \| ,$$

where $y = (y_1, y_2,...,y_k)^T$, $e_1 = (1,0_1,...,0_k)^T$ and $\delta_2 = \|f_0\|$, so we have $w^p = w^p_0 + \Sigma y_k v^p_k$.

Step 6: Calculation

$$B^p w^p = \bar{f}^p ,$$
$$f^p = e^p_0 - \bar{f}^p ,$$

and

$$\|f\| = \sqrt{\sum_{p=1}^{P} (f^p, f^p)} .$$

If $\|f\| < \delta_1 \times \varepsilon_1$ then we go to next step, or else let $w^p_0 = w^p$ and go to step 3.

Step 7: Calculation

$$A^p w^p = \bar{r}^p ,$$
$$r^p = b^p - \bar{r}^p ,$$

and

$$\|r\| = \sqrt{\sum_{p=1}^{P} (r^p, r^p)} .$$

If $\|r\| < \delta \times \varepsilon_2$ then the algorithm is stopped, otherwise we let $w^p_0 = w^p$ and go to step 3.

4.3. The Parallel N-S Solver

The method of distributed storage of the matrix data results in the corresponding geometric domain decomposition in solving N-S equations which is illustrated in Fig.5.

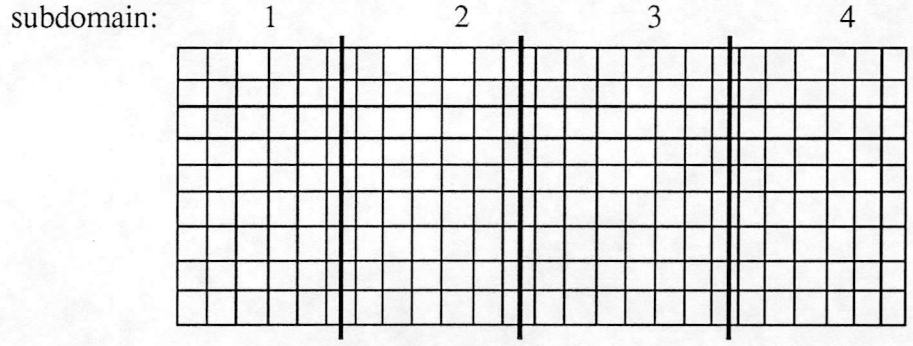


Fig 5. The domain decomposition.

The discretised N-S equations can be written as follows

$$F(Q) = 0 \tag{4.3}$$

The general Newton's method is

$$\left(\frac{\partial F}{\partial Q}\right)^n \Delta^n Q = - F^n(Q) \tag{4.4}$$

By forming $F(Q)$ and the Jacobian $J=\partial F/\partial Q$ at the known nth iterate, the increment $\Delta^n Q$ is then found by solving the linear system. The value of Q at the new iterate is given by

$$Q^{n+1} = Q^n + \Delta^n Q \tag{4.5}$$

In the parallel case the $F(Q)$ can be formed in each required subdomain.

The approach is to replace the analytic Jacobian with a numerically approximated Jacobian [5]. In the sequential computation case: Let h be square root of machine epsilon, if J is a band matrix of band-width $m=2\rho-1$ then the difference $F_i(Q+h\,e_j) - F_i(Q)$ is zero if $|i-j| \geq \rho$; it follows that we may find simultaneously approximations to columns j+km, k = 0,1,2,..., of J from the difference $F_i(Q+\Sigma\, h\, e_{j+km}) - F_i(Q)$. Here the sum Σ is for the subscript k. In this way the total number of subroutine calls needed can be reduced from n+1 to m+1. This strategy positively minimizes the total number of function evaluations in view of the number of unknown coefficients in each row of J. In the parallel case: Because the property of forming the Jacobian is according to columns the Jacobian J can be generated in each subdomain. This procedure should deal with a relative large subdomain compared with the original subdomain, which is divided according to the matrix storage.

After the Jacobian was generated in a parallel form we can use the parallel α-GMRES algorithm to form the completely parallel algorithm. The implicit iterative method begins from a relative 'good guess' and include three iterative loops: the inner GMRES iterative loop, the middle α-GMRES iterative loop, and the outer Newton iterative loop.

In the resulting developed parallel algorithm (described fully in [11]), communication is kept to a minimum and it leads to an efficient geometric domain decomposition type solver to complete this fully implicit overall approach. Fig 6 illustrates the convergence history using the Meiko M40 Computing Surface, and Fig 7 the speed-up achieved using 8 T800 transputer nodes providing a parallel efficiency of 83%. Fig. 8 demonstrates the reduction in memory size per node achieved.

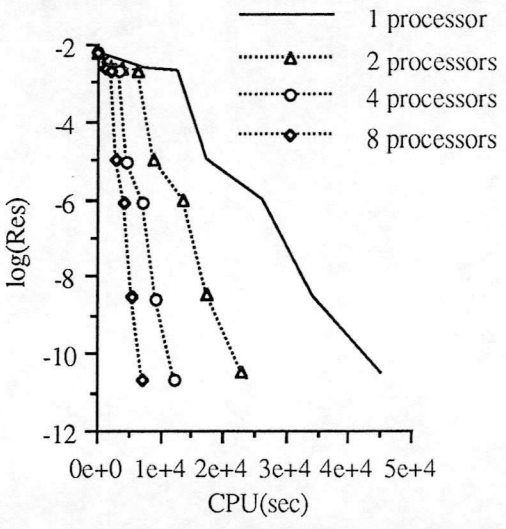


Fig 6. Convergence of N-S solution with different number of processors.

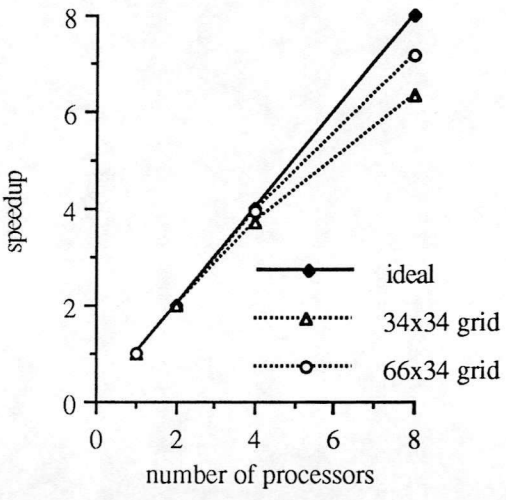


Fig 7. Speed-up acheived using 8 T800 transputer nodes.

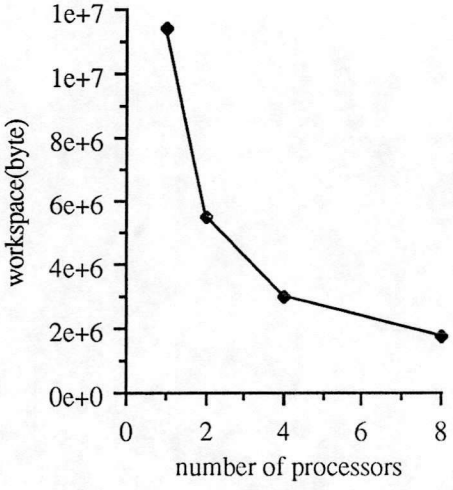


Fig 8. Memory needed for N-S solution with different number of processors

It is planned to extend the work to research the ability of the scheme to tackle more complex multi-dimensional problems using DL's Intel iPSC/860 Hypercube with faster processors and larger memory. The very preliminary work so far done has not brought the efficiency up to the level of that on the Meiko, because the processing on one i860 node is much faster than that on a T800 transputer. Steady state hypersonic and time-accurate unsteady transonic flow cases are planned in the future. The solver could have application in other areas.

5 CONCLUSIONS

Two studies of the parallelisation of algorithms to support the numerical calculation of hypersonic viscous flows over generic vehicle configurations by solving the Navier-Stokes equations have been made. High order Osher flux difference splitting upwind discretisation was used in each case to achieve high resolution of important shock and viscous phenomenon within the flow field. A scalable parallel explicit formulation on an Intel iPSC860 Hypercube computer was developed for calculating the 3-dimensional flow over a blunt delta wing at high incidence. This was demonstrated to achieve high parallel efficiency using up to 32 processors. A fully implicit formulation using a Newton-like procedure with a GMRES solver with pre-conditioning was applied to the calculation of the flow over a cone at high incidence. Again high efficiency was achieved using 8 transputers in a Meiko Computing Surface multi-processor. This implicit approach, although more complex, has potential in providing more rapid convergence characteristics, hence efficiency, than the explicit scheme and it is shown to be parallelisable. The work reported contributes to a programme of work to assist the development of computational tools to calculate accurately and efficiently steady and unsteady viscous compressible flows over complex aerospace configurations.

The first author was supported by University of Glasgow and ORS scholarship awards.

6. REFERENCES

1. N. Qin and B.E. Richards, Aeronautical Journal, 1991 pp 152-160.
2. N. Qin, Presentation to SERC CC-CFD Workshop, November, 1990. University of Glasgow Aero Report 9120.
3. R. Tracy, California Institute of Technology Report, GALCIT Memorandum 62, 1962.
4. N. Qin and B.E. Richards, Proceedings of the Workshop in Hypersonic Flows for Re-entry Problems, Part I, 1991, Springer-Verlag 1992.
5. N. Qin and B.E. Richards, Notes in Num. Fluid Mechanics, Vol. 29, Vieweg, 1990.
6. X. Xu, N. Qin and B. E. Richards, University of Glasgow Aero Report 9110. Int. J. of Numerical Methods for Fluid Dynamics (in press).
7. N. Qin, X. Xu and B. E. Richards, University of Glasgow Aero Report 9228. J. of Computing Systems in Engineering (in press).
8. N. Qin and B. E. Richards, Proceedings of the Workshop in Hypersonic Flows for Reentry Problems, Part II, 1991. (in press, Vieweg 1992).
9. J. Anderson, University of Glasgow Aero Report 9206, 1992.
10. Saad, Y and Schultz, M.H. *SIAM J. Stat. Comp.*, Vol. 7, No. 3, 1986, pp. 856-869.
11. X.Xu, N. Qin and B. E. Richards, University of Glasgow Aero Report 9210, April 1992. Journal of Parallel and Distributed Computing (under review).